

# 计算机网络课程设计实验实验报告

----- 基于中央定位服务器的 P2P 网络聊天系统设计

自 45 柳荫 2014011858

2016/12/23

## 需求分析：

聊天系统需要以下几个环节（模块）：


- 1，账号登陆上线，得与中央服务器通信，通知其你已上线；
- 2，P2P 下的一对一通信，这需要每个人既能充当服务器也能充当客户端；
- 3，查询好友是否在线，当你添加好友时，向中央服务器发出请求获知好友在线状态；
- 4，下线时得也得告知服务器你的状态；
- 5，文件传输，P2P 聊天时，让双方几次握手进行文件传输；
- 6，群聊功能，选中你的多个好友发起群聊，群聊时任何发出的消息需要广播到每个群聊者，是 P2P 的扩展。
- 7，友好的用户界面，要求程序编写者具有基本的界面编写功能，比如 java 中的 swing 包中的各种常用控件的掌握与运用。


## 总体设计&详细设计：


本人编写了 java 程序，源代码分为 4 个，也就是 4 个主类：**Test.java**, **Mainboard.java**, **MyClient.java**, **MyServer.java**。

## Test.java

Test.java 用于整个工程运行的第一步，即聊天系统启动界面，

 Log In





UserName	2014011858
UserPWD	.....
Server Ip	166. 111. 140. 14
ServerPort	8000

登陆

图形用户界面都是基于 **swing** 的，本想尝试 javaFX, 但由于临时更换大作业选项，时间有限，所以还是使用了传统的 swing。

简洁明快的登陆界面，采用了 BorderLayout 的布局管理。北边(BorderLayout.North)是一幅清新的图片，仿照了 QQ 的蓝天白云。西边(BorderLayout.West)是一个类似头像的标记，也是模仿 QQ 的企鹅。东侧面板未置放控件。中间面板(BorderLayout.Center)由一个 4\*2 的 GridLayout 布局构成，左边 4 个为 JLabel, 右边 4 个 JTextField 已经内置好相应的登陆信息，方便经常用同一个账号登陆，如果临时变了，也可以修改再登陆，非常方便。南边(BorderLayout.South)是一个大大的醒目的登陆按钮。

点击登陆按钮后，触发了一个事件响应，接着就如同那次 UDP 通信实验一样，向服务器端发信息，关键代码如下：

```

socket = new Socket(serverip, serverPort);

OutputStream toServer = socket.getOutputStream(); //输出流
toServer.write(messageSent.getBytes());
toServer.flush();

InputStream in = socket.getInputStream();           //输入流
String strIn;
byte[] answer;
while(true){
    if(in.available() != 0){                        //读到信息
        answer = new byte[in.available()];
        in.read(answer);
        strIn = new String(answer);
        break;
    }
}

System.out.println(strIn);
if(!strIn.equals("lol")){                          //不是 lol
    System.out.println(strIn);
    JOptionPane.showMessageDialog(null, "用户名或密码错误",

```

```
        "ERROR", JOptionPane.INFORMATION_MESSAGE);  
    }  
else{                                     //是 lol  
    MainBoard mainboard = new MainBoard(ID, serverip, serverPort);  
    dispose();  
}
```

如上代码，接收服务器返回的信息，如果不是'lol'则报用户名或密码错，否则登陆成功，进入主面板 MainBoard。

## MainBoard.java

MainBoard.java 用于登陆成功后的用户主界面：  
初始化界面一样的简洁明快：

上面的标志中，有自己的用户名(学号)，中间是用于将来  
放置同学（好友）信息的；下面是 5 个按钮。  
编写了多个事件驱动类：

```
/**
 * 添加好友按钮监听
 */
private class newFriendButtonListener implements ActionListener{...}

/**
 * 发起会话按钮监听
 */
private class P2PchatListener implements ActionListener{...}

/**
 * 群聊按钮监听
 */
private class groupChatButtonListener implements ActionListener{...}

/**
 * 删除按钮监听
 */
private class deleteFriendButtonListener implements ActionListener{...}

/**
 * 退出按钮监听
 */
private class leaveButtonListener implements ActionListener{...}

/**
 * 接收文件
 */
private class FileReceive extends Thread{...}
```



下面一一介绍：

### 窗口监听器 MWindowListener:

当关闭主界面窗口时，会向服务器发送相应信息如实验要求里讲的“logout”+本人学号，从而从服务器收到返回的信息 too, 然后在打印出一条信息 'I' ve logged out' :

```
Socket socket = new Socket(fitServerIP, fitServerPort);

OutputStream toServer = socket.getOutputStream();

String leaf = "logout" + myID;

toServer.write(leaf.getBytes());
```

```

        toServer.flush();

        InputStream fromServer = socket.getInputStream();

        String ACK;

        while(true){

            if(fromServer.available() != 0){

                byte[] answer = new
byte[fromServer.available()];

                fromServer.read(answer);

                ACK = new String(answer);

                //            System.out.print(ACK);

                break;

            }

        }

        toServer.close();

        fromServer.close();

        socket.close();

        if(ACK.equals("loo")){

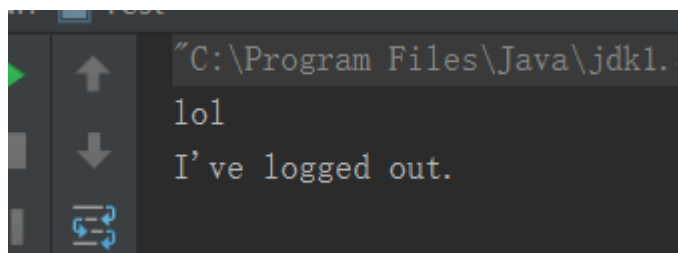
            System.out.println("I've logged out.");

            System.exit(0);

        }
    }
}

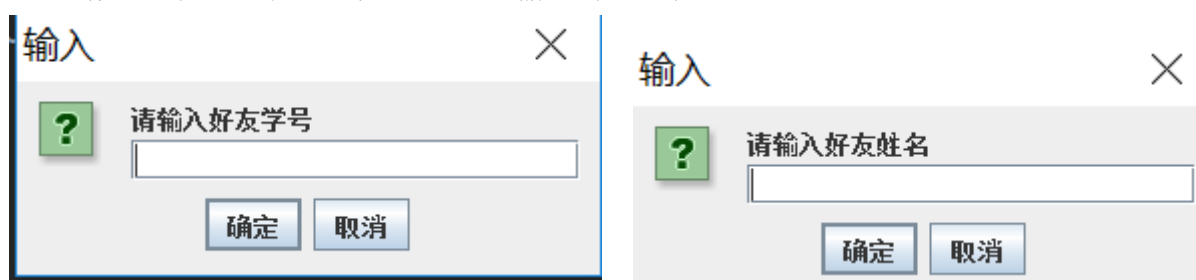
```

运行时控制台结果如下：



## 添加好友 NewFriendButtonListener:

会先后弹出 2 个对话框，让用户输入好友的姓名和学号：



当都输入了一些内容时，向服务器询问，测试一下输入的正确性以及好友的在线状态。这就又涉及到一个内部类 **TestConnect**，用于向服务器发

一个内部类 **TestConnect**，用于向服务器发

“q”+学号并返回服务器的信息，然后回到

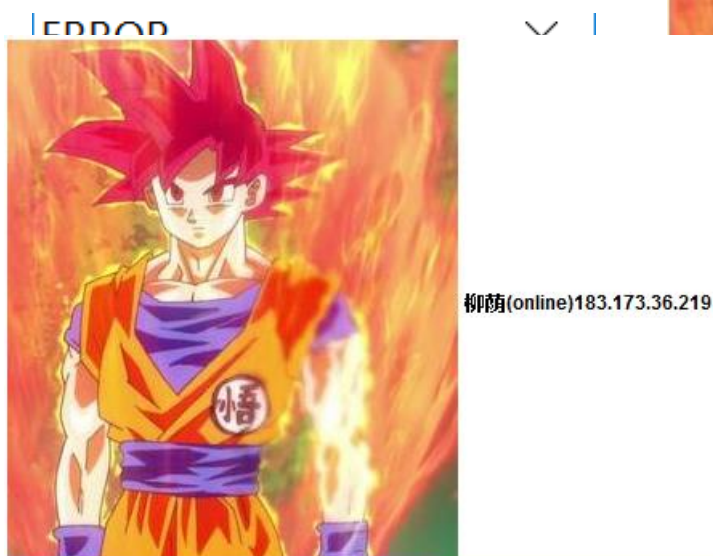
**NewFriendButtonListener** 类，根据返

回信息不同采取不同的处理办法，若返回

‘n’，则显示 **offline**；若用户名不对（不是

选计网课的学生的学号），就报错；否则在线，

显示 **online** 并显示 ip，截图分别如下：

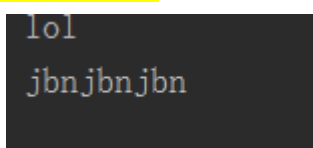


### 删除好友 deleteFriendButtonListener:

选中一个或多个好友，点击删除，将其从列表中删除。

### 退出按钮 leaveButtonListener:

点击退出，向中央服务器发送 'logout' + 本人学号，若系统返回 loo，则成功退出，打印 'jbnjbnjbn'。截图：



```
lol
jbnjbnjbn
```

### 发起会话 P2PchatListener:

选中好友，若好友在线，则调用一个新的客户端类，开启新的会话，详见后文介绍。

### 群聊按钮 groupChatButtonListener:

同上选中至少 1 个好友，当选中好友全部在线时可以发起群聊，弹出新的群聊界面，同样详见后文介绍。

### 接收文件 FileReceive:

其中有处理接收到的文件的实行 Runnable 的内部类，进行文件的打包接收，同样详见后文的 MyClient 和 MyServer 的分析。



## MyClient.java & MyServer.java:

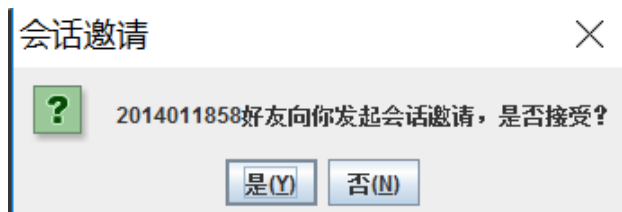
这 2 个程序用于实现真正的聊天、群聊以及文件传送等功能。

率先发起通信的是作为客户端的，首先在主界面中选择了好友然后点击“发起会话”按钮后，会弹出一个聊天界面：



这个界面风格精简，最上方有一个聊天图标，主体部分是 2 个滚动面板，下方用来草拟要发的信息，上方用来显示已经聊的内容，右下角有 3 个功能按钮“发送消息”，“传送文件”，“关闭窗口”。

此时 TCP 连接已经建立。同时作为被邀请的服务器方，会收到客户端发来的消息 `userID + "single_p2p"`，会显示一个确认对话框，当选择“否”时，客户端自动告知被拒绝对话，然后关闭输入输出流，并断开 TCP 连接，服务器端也是如此；选择“是”时，服务器端也建立相应的聊天面板，开始聊天。



## P2P 聊天建立

关键代码如下：

客户端:

```
//建立 TCP 连接

socket = new Socket(studentIP, 49876);    //Socket 内容是好友 IP 地址 +
端口号 49876

//作为客户端首先建立输出流

P2POutput = socket.getOutputStream();
P2PPrintWriter = new PrintWriter(P2POutput);

//获取输入流

P2PIn = socket.getInputStream();
P2PBuff = new BufferedReader(new InputStreamReader(P2PIn));

//发送通话请求

P2PPrintWriter.write(userID + "single_p2p" + '\n');
//single_p2p 是建立单独聊天请求的标志

P2PPrintWriter.flush();

//接收对方的许可

String confirm = P2PBuff.readLine();
System.out.println("confirm" + confirm);
if(confirm.equals("p2p_cancel")){           //如果对
```

方拒绝此次通话

```

JOptionPane.showMessageDialog(null, "对方拒绝此次会话", "SORRY",
JOptionPane.INFORMATION_MESSAGE);

try{
    //关闭输入输出流，断开 TCP 连接
    socket.close();
    P2PBuff.close();
    P2PIn.close();
    P2PPrintWriter.close();
    P2POutput.close();
    dispose();
} catch(IOException ignored){
}
}

else{ //如果对方允
许此次通话

```

服务器端：

```

//作为服务器，首先获取输入流
in = socket.getInputStream();
buff = new BufferedReader(new InputStreamReader(in));

```

```
//获取输入流之后再建立输出流

output = socket.getOutputStream();
printWriter = new PrintWriter(output);

//对方在发起会话时首先会发送一个会话请求"single_p2p"，只有当用户同意会话请求时，才建立通话对话框

String require = buff.readLine(); //接收好友的会话请求

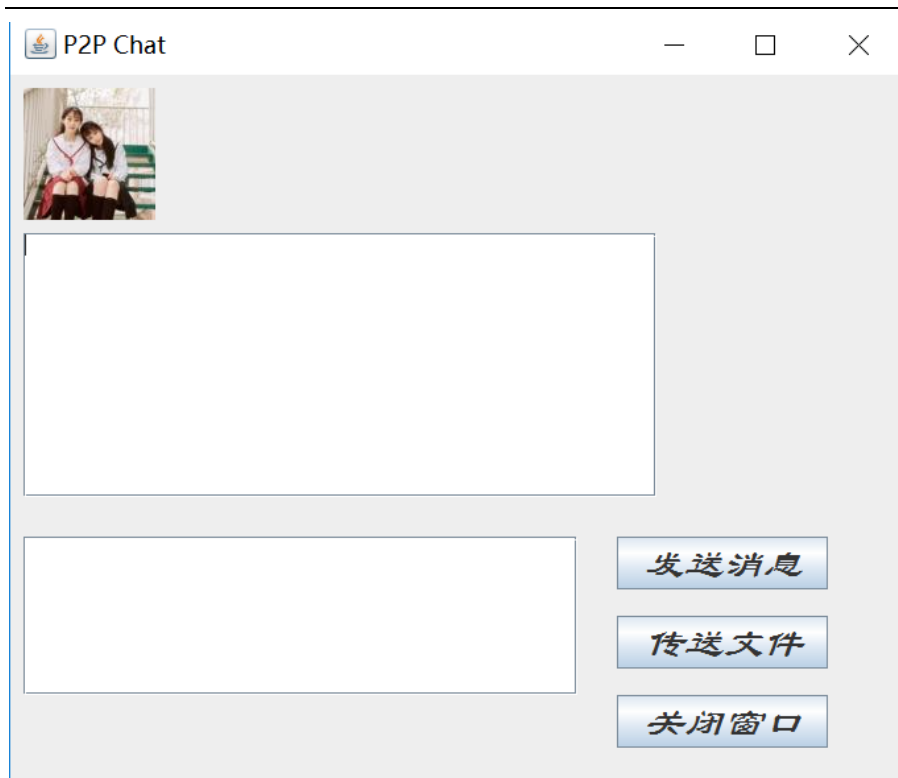
if(require.contains("single_p2p")){ //如果是单独聊天

    //弹出确认对话框，让用户选择是否接受会话邀请

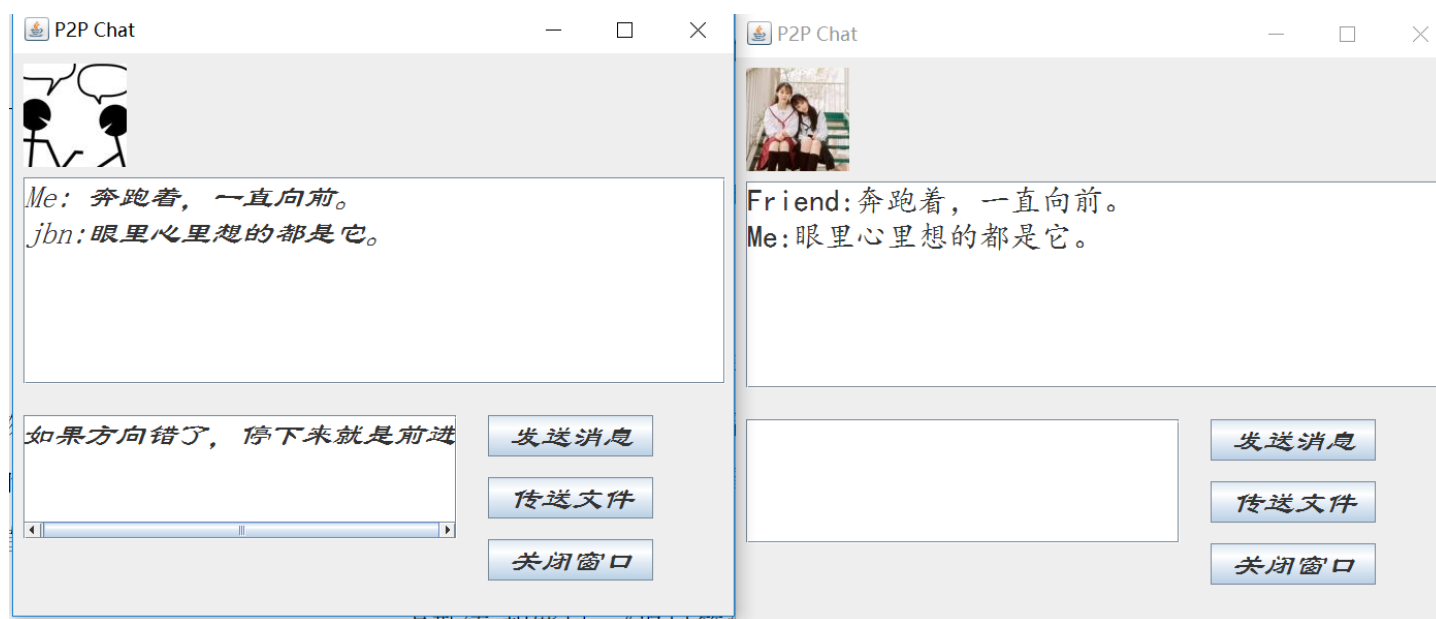
    String sub = require.substring(0, 10);

    confirmP2P = JOptionPane.showConfirmDialog(null, sub + "好友向你发起会话邀请，是否接受？", "会话邀请", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
}
```

服务器端同意聊天之后会建立聊天面板，同样简洁明快的风格：



下面可以开始愉快的聊天了：



## P2P 聊天消息发送与接收处理：

这都是在客户端和服务端的消息发送按钮监听以及对输入流的处理 2 个类中实现的，

以服务器端的为例：

```
/**
 * 消息发送按钮监听器
 */
private class messageSendButtonListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
        //获取需要发送的文本，并且在末尾加上换行符
        String send = textArea2.getText() + '\n';
        //写到输出流中
        printWriter.write(send);
        printWriter.flush();
        textArea1.append("Me:" + send);
        textArea2.setText("");
        textArea2.setFont(new Font("楷体", Font.ROMAN_BASELINE, 30));
        textArea1.setFont(new Font("楷体", Font.ROMAN_BASELINE, 30));
    }
}
```

```
else{
    //若不是文件发送的相关信息，则作为聊天内容显示在文本域中
    textArea1.append("Friend:"+receive+'\n'); //最常见的信息发送
    textArea1.setFont(new Font("楷体", Font.ROMAN_BASELINE, 30));
}
```

分别是其消息发送和接收的关键代码，客户端也类似。

## P2P 聊天结束通话：

下面介绍 2 端的结束通话的实现：可以点击右上角的按钮，也可以点击“关闭窗口”按钮，然后向对方发出一个 `socket_close` 的讯息，同时关闭窗口，断开 TCP 连接；对方接收后得到告知，并关闭窗口，断开 TCP 连接。



## P2P 聊天文件传输

P2P 聊天时，有文件传输功能，

```

/**
 * 文件传输按钮监听器
 */
private class fileSendButtonListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
        //首先向对方提出发送文件的请求
        printWriter.write("I_want_send_file" + '\n');
        printWriter.flush();
        JOptionPane.showMessageDialog(null, "正在等待对方确认，单击“确认”继续等待", "WAIT", JOptionPane.INFORMATION_MESSAGE);
    }
}

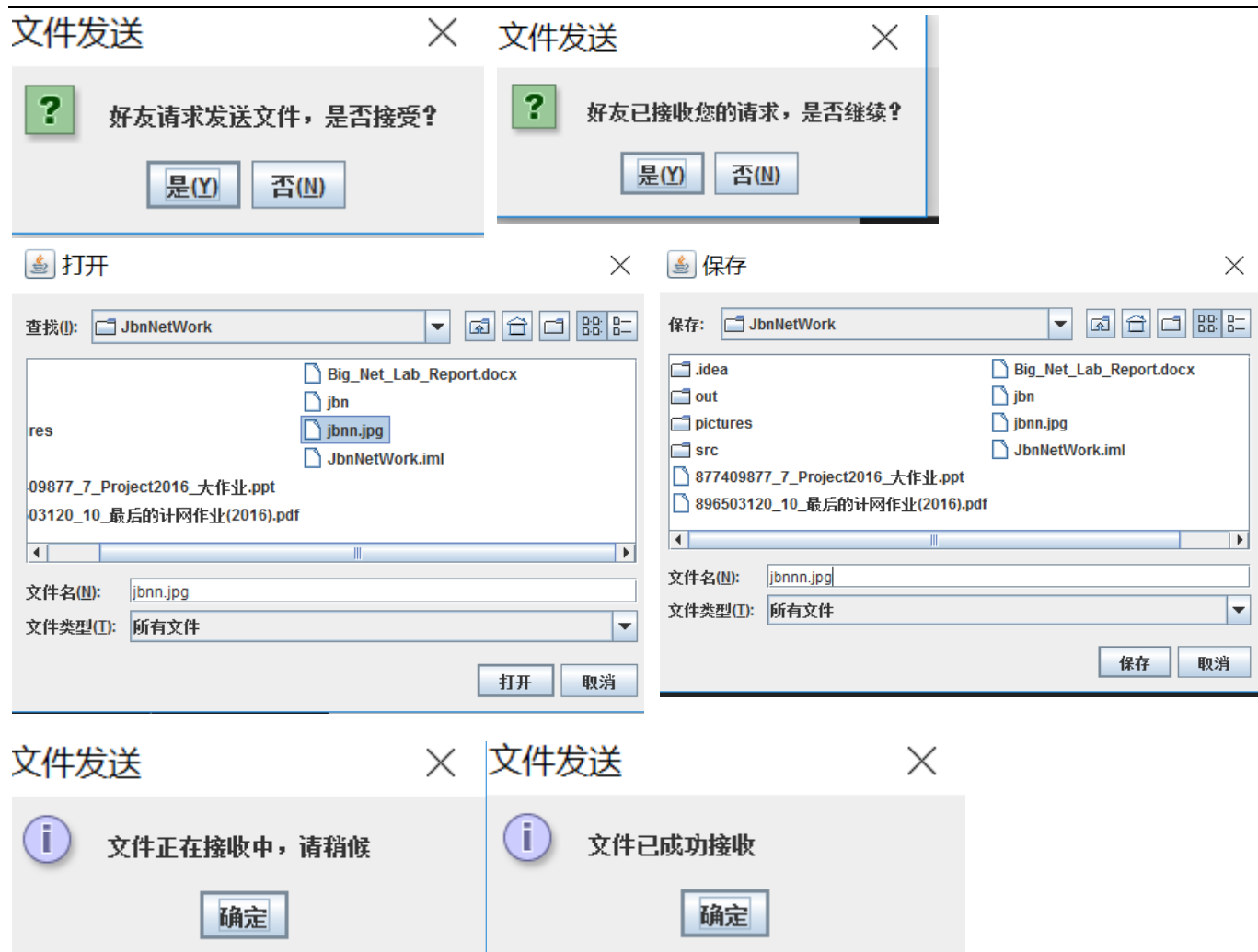
//需要判断是否是发送文件的请求，如果不是，则视为聊天内容显示在文本域中
if(!(receive = buff.readLine() == null)) {
    if(receive.equals("I_want_send_file")){
        //作为文件的接收方，对方请求发送文件
        //Client在文件发送按钮响应器中发送了这个字符串
        int confirmFile=JOptionPane.showConfirmDialog(null, "好友请求发送文件，是否接受？", "文件发送", JOptionPane.YES_NO_OPTION);

        if(confirmFile == JOptionPane.YES_OPTION) { //允许好友发送文件
            printWriter.write("I_agree" + '\n');
            printWriter.flush();
        }
        else{ //拒绝好友发送文件
            printWriter.write("let_down" + '\n');
            printWriter.flush();
        }
    }
    else if(receive.equals("I_agree")) {
        //作为文件的发送方,对方允许发送文件
        //好友已经允许用户发送文件的情况下，用户是否继续发送文件
        int confirm = JOptionPane.showConfirmDialog(null, "好友已接收您的请求，是否继续？", "文件发送", JOptionPane.YES_NO_OPTION);

        if(confirm == JOptionPane.YES_OPTION) { //若用户继续发送文件,开始一个新的线程发送文件
            FileSend fileSend = new FileSend();
            new Thread(fileSend).start();
        }
    }
    else if(receive.equals("let_down")) {
        //作为文件的发送方,对方拒绝发送文件
        JOptionPane.showMessageDialog(null, "好友拒绝了您的发送请求", "文件发送", JOptionPane.INFORMATION_MESSAGE);
    }
}

```

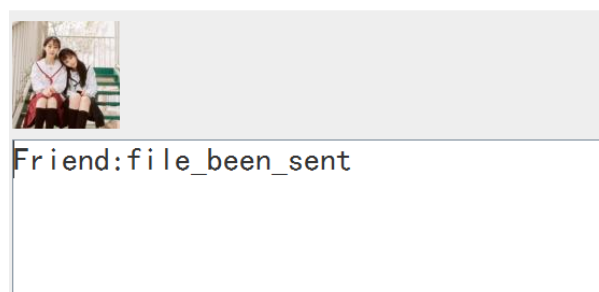
上面是关键核心代码，其一是“发送文件”按钮监听器，其二是与文件相关的多次握手协议，服务器与客户端的代码基本一样，都是 A 先发送请求，B 同意的话，再问 A 是否继续发送，A 同意的话，则选择要发送的文件，发送给 B，B 选择保存位置，将其接收并保存。



打开文件夹，发现里面多了这个文件。

 jbnnn.jpg	2016/12/22 23:41	JPG 文件
--	------------------	--------

且接受文件的那一方收到了发送方的信息 `file_been_sent`:



当然上述功能的实现还涉及到另一个类：文件发送类 FileSend:

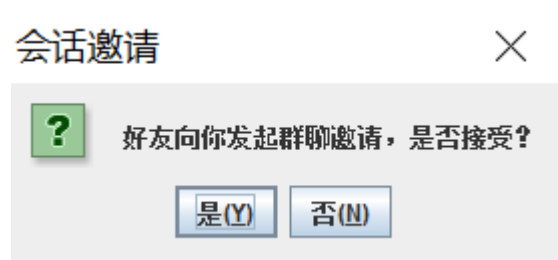


```
/**
 * 文件传输类，继承父类Thread，与实行Runnable方法一样，可作为线程运行使用
 */
private class FileSend extends Thread{
```

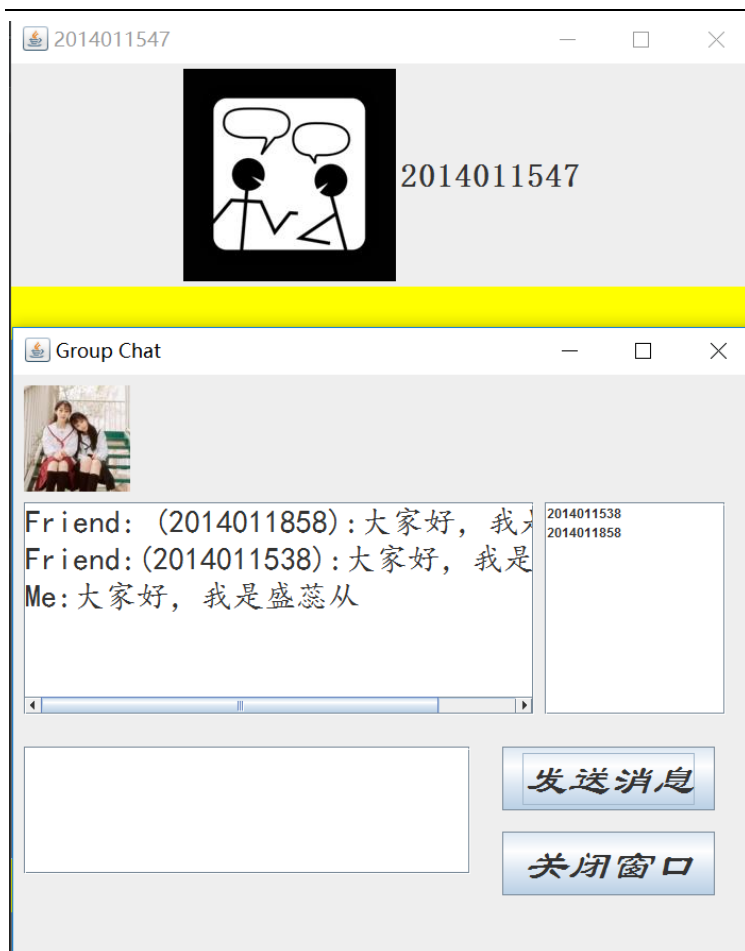
首先建立 TCP 连接，然后建立输出流，然后用 JFileChooser 类的对象方法选择要打开的文件，建立输入流读入文件，建立输出流，以字节为单位发送，最后关闭输入输出流，断开 TCP 连接。

### 群聊功能：

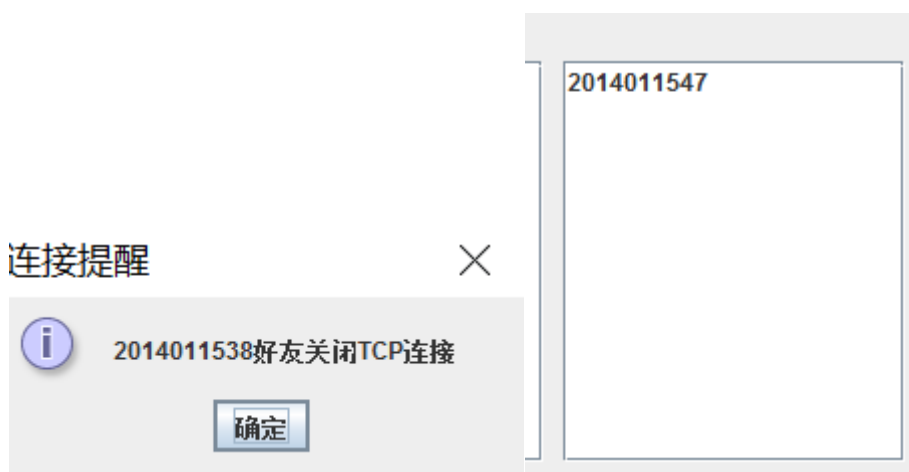
先选择至少一个好友，在主面板上点击“群聊”按钮，则其充当客户端，向同学发送群聊邀请信息，若至少 1 人同意，则可以群聊，将参与群聊的人的学号广播到全体参与群聊的人，然后所有参与群聊的人都启动了聊天界面：







右侧面板可以显示实时群聊者的学号（除自己）。当有人“关闭窗口”时，会广播到其他人，并在右侧学号中去除。



另外，当群聊倒数第二人退出后，最后一人会被告知其他人都退出群聊，然后自动将其群聊 TCP 连接断开，群聊界面关闭。

## 结果分析(遇到的问题 and 原因)：

### 问题 1：

原来对 java 中的多线程还没有很深的了解和认识，导致简单功能的实现出现了 bug，运行不起来。

### **解决办法：**

使用多个内部类，每个类或者 `implements Runnable` 或者 `extends Thread`，以便作为线程不断运行，这样在主类中使用不同类别的对象时（这通常是监听器对象），就能同时启动多个线程，进而使程序能够非常协调地工作。

### 问题 2：

群聊功能的实现时好友的动态管理。

### **解决办法：**

用传统的数组感到十分地不便，尤其在删除一个元素时；于是采用了一种对象集合 ArrayList 的数据结构分别存放群聊同学的 IP、ID、name 等信息，这样管理起来就感到比较容易。

## 总结：

这次大作业使得我们把计网课学到的知识应用到了实际当中，对于线程、端口、套接字等等概念有了直观和实际的认识，综合性非常强。聊天软件是一个我们日常时时接触又非常有意思的应用，我们在编写的过程中也收获了很多乐趣，这样一个程序对我们来说已经有点接近“作品”了而不是作业。

另一方面，我也更加深入地理解和学习了 java 语言中的进阶一些的知识，能比较完整地理解其 socket 通信。由于本人网络学堂上作业截止日期是 1 月 1 日，而原来的 duckietown 小车项目进度并不快，我感到难以在日期内完成，所以临时变更了大作业，仓促之中编写的程序，虽然有些细节地方比如 JFileChooser 那里也向学长请教了代码的编写，但总体上自己还是比较满意的，自己也锻炼了很多。总而言之，这次计网大作业让我逐渐体会到了一个大项目的协调与架构的不容易。虽然在写的过程中遇到了很多问题，但同时也学会了很多调试方法与调试技巧。

## 参考文献：

- 计算机网络—自顶向下方法. James.F.Kurose, Keith.W.Ross.
- Introduction to Java Programming—Comprehensive Version. Y.Daniel Liang
- Java 语言程序设计. 谌卫军

## 注，选做内容：

- a) 友好的用户界面.
- b) 群聊功能.